Integrated software environment for building design and construction

S J Fenves*, U Flemming†, C Hendrickson*, M L Maher* and G Schmitt‡

A prototype system, the Integrated Building Design Environment (IBDE), has been implemented to act as a testbed for a number of issues that arise from the need to integrate computer aids for the design and construction of buildings. The seven programs that comprise IBDE are briefly described. Issues such as knowledge representation, data organization, intercommunication, implementation and control are discussed.

The design and construction of buildings should be viewed as complementary and mutually dependent processes1. Design is the process of creating the description of a facility, usually represented by detailed plans and specifications, while construction consists of performing the activities and applying the resources required to realize the design envisioned by the architects and engineers. Effective communication, information management and decision-making among the various organizations responsible for design and construction are essential to ensure quality and economy in the final product. While communication among different specialist organizations is a particular problem for construction, even turnkey projects designed and constructed by a single organization can have major problems with information flow and integrated decision-making. Effective integration can result in significant cost and time savings, especially in complex projects2.

An important development encouraging integration is the increased effectiveness and reduced cost of computer aids. As computer aids are used more extensively in both design and construction, computer based environments become increasingly attractive to improve communication among different professionals and organizations. But integrated environments are difficult to develop and implement, raising fundamental issues in the design/construct process. These issues include the following:

- Multiple disciplines
  - The integration of design and construction involves professionals from many different disciplines, which introduces discrepancies in terminology, attitudes, styles etc.
- Multiple views of the building
  - Each discipline has its own perspective of the building. For example, a structural engineer is interested in the size and location of structural elements such as beams and columns, while an architect is interested in the character and expressiveness of the building as a whole as well as in the function and appearance of its spaces and enclosures. Maintaining all views of the building introduces problems of redundancy and consistency of information.
- Poor communication
  - Currently communication occurs verbally or through the medium of design drawings. Intermediate decisions and justification for decisions made during design or construction planning typically are not communicated.
- Varying degrees of automation
  - Each discipline uses computer programs to assist in its tasks. The preparation of input data and interpretation of output are typically done manually.
- Incompatibility of hardware and software
  - The hardware and software that an organization uses may not be compatible with that used by other organizations. This makes it difficult to use existing software within an integrated environment.

The Integrated Building Design Environment (IBDE) is a prototype intended to serve as a testbed for examining these issues. Its demonstration and test domain is the architectural planning, structural design and construction planning of high-rise, speculative office buildings. Within this context, it addresses general communication and control issues. Attention is focused, in particular, on two aspects: (1) the representation and communication of the project information as the project progresses; and (2) the control of the overall process.

*Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
†Department of Architecture, Carnegie Mellon University, Pittsburgh, PA 15213
‡Formerly with Department of Architecture, Carnegie Mellon University, Pittsburgh, PA 15213, now at ETH, Zurich, Switzerland

IBDE currently integrates seven independent computer programs. Some of these programs existed before work was started on IBDE (the existence of these programs, in fact, provided a strong initial stimulus to start the project), while others were developed specifically for IBDE. All programs are knowledge-based systems using declarative or rule-based representations of knowledge that permit rapid development and modification. The knowledge-based processes currently in use are the following:

- ARCHPLAN: a front end for the interactive development of a design concept
- CORE: a space planner for the service core
- STRYPES: a structural system configurer
- STANLAY: a structural layout and approximate analysis system
- SPEX: a structural component designer
- FOOTER: a foundation designer
- CONSTRUCTION PLANEX: a construction planner, estimator and scheduler.

Integration in a commercial environment would necessarily include a number of conventional programs.

Communication between the processes is mediated by a controller, a separate process that relies on two mechanisms: one is a message blackboard that is used to communicate project process information and the other is a project data store maintained by a data manager that records the information generated and used by the processes. The controller uses the information posted on the blackboard to initiate the execution of individual processes. It also directs the data manager to provide and receive the information shared between the processes. As the different processes reside on different machines, the data manager and the blackboard rely on a local area communication network. The overall organization of the environment in its first and pilot implementation is shown in Figure 1.

IBDE contrasts sharply with the integrated systems in use or under development in the building industry. However, the principal contrast is not between knowledge-based or algorithmic process components. Integrated systems in industry achieve a high level of data integration by tying CAD systems, analysis programs, etc. together through a shared database. But such systems do not address the integration of building systems, of design and construction processes and of the thought processes of the disciplines involved. These are the issues addressed by the IBDE project, for which the content and semantics of the communication are as important as the mechanisms of data transfer.

The remainder of this paper describes the environment. The next section describes the seven processes in terms of their function and contribution to the evolving design description or to construction planning. The third section outlines the global representation of design and construction information. The fourth section describes control and communication in IBDE. User interfaces are discussed in the fifth section, and the sixth section provides a summary.

**PROCESSES**

A brief description is given of the processes that emphasizes their generative nature. Interested readers may find more detailed descriptions in the references.

**ARCHPLAN**

ARCHPLAN is a knowledge-based ARCHitectural PLANning expert system intended to assist in the development of a design concept. The input describes the site, the client’s programme and budget, as well as applicable geometric constraints. The output provides 3D information on the overall geometry of the design; the distribution of functions within this geometry; and major circulation elements.

The conceptual design process is subdivided into three corresponding decision modules that are distinct, but interrelated. ARCHPLAN establishes at the beginning a generic prototype that can be modified by the user in combination with the knowledge built into the program. The design process starts with the site, cost, and massing module that derives a massing model satisfying the given budget and a range of other parameters. Cost, site and massing options are interdependent concerns. The user describes the degree of commitment to a certain requirement, such as floor-to-floor height or ground-floor area, by entering a certainty factor. Conflicts are resolved based on these factors. Simple optimization options are also available: minimum cost, maximum daylighting, or a combination of the two.

The function module determines the vertical and horizontal distribution of architectural functions within the basic massing scheme (office, retail, atrium, mechanical and parking space). The module proposes a 3D layout which is displayed in solid or wireframe representations.

The circulation module allocates the major elements of vertical circulation. They have a major impact on...
the internal organization and on the architectural expression of a high-rise building. The two extreme cases are the internal (service and elevator core in the centre of the building) or the external solution (service and elevator cores attached to the outside of the building). The module creates circulation proposals based on variations of these two prototypes.

CORE

CORE generates layouts of the elements in the service core of the building under consideration (elevators, elevator lobbies, restrooms, emergency stairs, utility rooms etc.). It is an adaptation of LOOS, a general system for the generation of layouts in various domains. LOOS places particular emphasis on the generation of layout alternatives with interesting trade-offs. The first version of CORE consists of the following modules.

Pre-processor

This module accepts basic data describing the overall geometry of the building under consideration and the anticipated size and location of the service core. Based on this information, the module determines an optimal banking arrangement for the elevators needed to serve the floors of the building, namely: the number of banks; the number and speed of cars in each bank; and the floors served by each bank. It also determines a list of ancillary areas to be included in the core (emergency stairs, restrooms, utility rooms, etc.) These rooms, together with the elevator banks, are the objects to be allocated in the following module.

Layout designer

A level of complexity is added to the layout problem if more than one elevator bank is needed. In this case, the layout of the core changes for the different sections of the building served by different banks. Alternative layouts might exist for each section, but the layouts selected for each section must be compatible with each other. We plan to handle this situation through a stepwise process that produces sequentially:

- layout alternatives for all banks on the first floor which fixes the position of the hatchways, cars and lobbies in each section
- layout alternatives for each section that are compatible with a selected first floor layout
- a combination of layouts for each section that are compatible with each other.

The first version of the system handles only single banks and designs the layout of the core on a typical floor.

Post-processor

This module selects one of the layout alternatives generated by the previous module and finalizes it, for example, by adding structural elements such as load-bearing walls.

STRYPES

STRYPES is a knowledge-based expert system that configures a structural system for the building under consideration. It is based on knowledge acquired through the development of HI-RISE and implemented in EDESYN, an expert system shell for engineering design synthesis. The input to STRYPES includes: a structural grid produced by ARCHPLAN, specifying potential locations for structural systems; functional information about the building, such as intended occupancy and location and size of the service core; and load information. The output specifies system types and materials for the lateral load system and for the vertical and horizontal gravity load systems.

The configuration process is decomposed into the design of the lateral load system and the gravity load system. The lateral load system is described by a 3D lateral system, such as a core, tube, or orthogonal 2D systems; a 2D lateral system in each direction, such as a braced frame, rigid frame, or shear wall; and a material, such as reinforced concrete or steel. The constraints that guide the generation of alternative lateral systems consider architectural, structural and construction feasibility. For example, shear walls are not considered for office buildings unless they are placed around the service core.

The gravity load system is described by a 2D horizontal system (such as a reinforced concrete slab, corrugated steel topped with concrete, prefabricated concrete panels, or a waffle grid) and by the support conditions (such as the number of edges supported and the existence and direction of intermediate floor beams). The selection of a gravity system is based on compatibility with the architectural layout, the lateral system and construction requirements.

STANLAY

STANLAY, also developed using EDESYN, performs two major tasks. First it lays out the structural systems specified by STRYPES. The location of the lateral load systems requires the specification of the number of 2D vertical systems, such as rigid frames or shear walls, and their location on the given grid. The knowledge-base for STANLAY contains layout alternatives for each type of lateral load system. Each layout, once it has a specified grid location, is checked for feasibility before the next task is performed. A single layout is selected, based on efficiency of material use under the assumption that this is related to cost.

The second task is an approximate analysis of the lateral and gravity load systems to determine the load requirements for component design. The components of the structural systems are grouped according to similarity of function and location. The output of STANLAY provides a definition of components and their required load capacities.

SPEX

SPEX performs the preliminary design of structural components for the structural system specified by STANLAY. It implements a strategy in which components are designed automatically by applying three types of knowledge: knowledge contained in design standards; ‘textbook’ knowledge of structural, material and geometric relationships; and designer-dependent expertise. The latter knowledge consists of rules for
generating a design focus, which identifies one or more behaviour limitations that the designer hypothesizes as governing the design of a specific component.

This design strategy is implemented in the following modules:

- Design focus generation: the designer-supplied rule base is used to select a design focus.
- Requirement retrieval: the requirements of the named standard corresponding to the design focus are retrieved.
- Constraint set generation: the constraints resulting from the requirements are assembled.
- Constraint set satisfaction: an optimal solution to the constraints is generated using either mathematical optimization or database lookup.
- Conformance verification: in the case of detailed design the solution based on the initial design focus is verified against all requirements pertaining to the component. If some requirements are violated, SPEX backtracks either to generate a new set of constraints or a new design focus.

The output of SPEX describes the optimal components (e.g., square concrete column 12 in wide with reinforcement ratio of 0.025, or steel beam designation W 12 × 84). In the IBDE implementaton of SPEX, two shortcuts have been introduced for efficiency: first, constraint set satisfaction is performed by table lookup from pre-generated tables of optimal component configurations; and second, only preliminary design is performed, so that conformance verification, and thus the possible need for backtracking, is omitted.

**FOOTER**

FOOTER is an expert system that performs a preliminary design of the foundation of the given building. It is also implemented in EDESYN. The input to FOOTER includes: soil conditions, such as the presence of obstructions, location of water table, depth of bedrock, and soil classification, and imposed load conditions from the structure provided by STANLAY. The output of FOOTER is a description of a footing or pile for each column and/or shear wall.

The problem of designing a foundation is decomposed into several subproblems: selection of foundation type, material type, casting type, excavation type, and parametric design of the foundation. A solution is determined by a combination of selections for each subproblem. The constraints in FOOTER include heuristics on the appropriateness of a specific foundation design for the given input conditions. For example, a timber pile is not appropriate if the column load exceeds 270 kips. Other constraints eliminate alternatives that are inconsistent combinations of partial solutions.

**CONSTRUCTION PLANEX**

CONSTRUCTION PLANEX is a knowledge-intensive expert system intended to assist the construction planner. The input to the system consists of specifications of the physical elements in the design provided by other processes, which may be grouped by section; site information (such as soil type and elevations); and resource availability (such as number of crews or equipment types). The output consists of a complete plan of construction activities including a provisional schedule and cost estimates.

In the initial creation of a construction plan, CONSTRUCTION PLANEX performs the following sequence of operations:

- Create element activities for design elements. This operation identifies a set of element activities required to construct each design element.
- Group element activities of common characteristics to give a hierarchy of element activities similar to that of MASTERFORMAT.
- Determine amounts of work for element activities. Geometric information for the quantity take-off is inherited from design element frames in the central data store.
- Select units of measure for element activities.
- Determine material packages for element activities based on design specifications.
- Create project activities that aggregate element activities and provide summary information on the underlying element activities.
- Select technologies for project activities. Technologies are chosen at a macroscopic or project level since consistency in this regard will reduce costs. Structuring activities into a hierarchy facilitates this imposition of overall constraints on the construction plan.
- Determine precedences for project activities. In CONSTRUCTION PLANEX, scheduling is performed at the project activity level, reflecting the homogeneity of resource use and the small granularity of detail contained in the underlying element activities.
- Compute lags for project activities. Element activities of several project activities are structured into an element activity subnetwork. A simple critical path algorithm is used to determine relevant lags among project activities based on this subnetwork.
- Estimate durations for project and element activities. This estimation process relies on a hierarchical decomposition similar to that described by Hendrickson et al.
- Schedule project activities using CPM, resource allocation and constraint satisfaction.
- Estimate costs by computing activity costs and project costs using unit costs and scheduling information.

**Evolution of processes**

The processes in IBDE are currently being extended both to generate and to react to criticisms and feedback. The initial version of IBDE operates automatically in the ‘forward pass’ mode, where the controller activates the processes in a linear, sequential fashion. The addition of CORE introduced some parallel processing, where STRYPES and CORE could be running simultaneously. The addition of CORE also introduced potential conflicts; for example, the preprocessor of CORE can find the core area set aside by ARCHPLAN inadequate, thus invalidating
the design produced by STRYPES and STANLAY, which is based on the same area. Similar conflicts can arise at any point in the design and construction planning process. Furthermore, a 'downstream' process may provide feedback on ways that the design could be improved.

The introduction of criticism and feedback in IBDE begins to address and identify potential conflicts. The strategy that we adopt for adding design critics takes advantage of the processes that already contribute to the project. Each process is extended to include a process activator and a process critic, as illustrated in Figure 2. The process critic posts one or more constraints on the message blackboard if the process was unable to produce a valid solution or if it can suggest possible improvements. The process activator serves two purposes:

- to produce criticism by checking the input data for scope; if the input includes design decisions that fall outside the capability of the process a constraint is posted on the blackboard and the process is not activated.
- to react to criticism by adding a constraint for consideration by the process; when the process is run again, the constraint ensures that the same conflict does not arise.

These two extensions allow criticism and feedback from the individual processes to be communicated to the controller in the form of constraints. Thus far these extensions appear to be easily incorporated into the original knowledge-based processes.

The authors also plan to include specialized design critics in IBDE. These programs are intended to evaluate design decisions using knowledge external to the original knowledge base that produced the decisions. Currently under development is a structural analysis critic that will execute a formal analysis using the results of SPEX and will check for consistency with STANLAY's approximate analysis.

GLOBAL REPRESENTATION

Objectives

The project data store holds the global representation of the building and serves as the repository of data communicated between the IBDE processes. The initial design and subsequent evolution of this data store were dictated by four considerations:

Provision of process views. The primary function of the data store is to provide individual views or subschemas to each process without regard for where this information was generated, and to transmit its output, again without regard for where it may be subsequently used. These views need to be sufficiently flexible so that changes in the processes' data needs can be readily accommodated.

Flexibility of global schema implementation. In contrast to the process views, which must be highly responsive to the processes' functional needs, the global schema - known only to the data store manager - can have an evolution of its own. As discussed below, a migration has successfully been made from a tabular, flat file organization to a relational DBMS organization.

Explicit representation of important conceptual relations. In a static environment, the data store need not contain more than the union of the processes' input and output requirements; all other data may reside inside the processes, following the principles of information hiding and data encapsulation. As a dynamically evolving environment was expected, it was decided to include in the data store important conceptual relations and abstractions able to facilitate the subsequent addition of new processes and critics. Thus, for example, the objects representing structural functions and frames are presently used only by STRYPES and STANLAY; their successor processes (SPEX, FOOTER and CONSTRUCTION PLANEX) deal only with individual building elements, such as beams or columns. Nevertheless, these high-level objects are included in the data store for subsequent use.

Support for common display interface. The visualization of an object as complex as a building requires the display of a great variety of information. Rather than delegating this display to the individual processes, a single common display interface was desired, which could display all the information in the data store in a variety of formats.

Overall organization

The data store is hierarchically organized as a tree of
related objects. Objects may represent very high-level abstractions, such as the entire building, or very detailed information, such as individual building elements. The hierarchy primarily represents part-of relations, where each object is a part or component of a higher-level parent object. Provisions are also made for representing is-alternative relations, where an object is an alternate design solution of the parent object. Through this latter relation, redesign in response to critiques received is readily supported. The data store provides at any time a complete snapshot of the current state of the design and construction planning process.

This overall organization is independent of the internal global schema implementation. Figure 3 shows the hierarchical organization among the major object classes in the current (relational DBMS) implementation. With this organization, each process can view the contents of the data store relevant to it, but not of the segments relevant to the other processes. This organization has supported the concurrent development of the processes and provides complete data and process independence among the processes.

**Communication with processes**

Communication with the processes is the responsibility of the data store manager. It works in concert with the controller and is responsible for supplying input data to the processes and retrieving their output data. Before initiation of a process by the controller, the data store manager transfers the input data to the machine on which that process resides. When a process terminates, it leaves its output on its own machine; when its termination message is received, the controller causes the data store manager to retrieve the data from that machine and merge it into the data store.

The data store manager also generates views or subschemas as needed by the processes, including all format and structural conversions. It is further responsible for maintaining an ‘audit trail’ in the sense of maintaining two descriptions for each object class: placed-by (the name of the process which provides the value(s) of its attributes(s)); and used-by (the names of the processes which use these values). Using this information, the controller can route any critique of a particular object to the process responsible for creating the object’s attribute values.

**Implementations**

The local view of each process consists of sets of objects with attributes in the implementation language of the process. Furthermore, in most processes, no explicit distinction is made between input and output attributes; the object contains all necessary attributes.

In the initial implementation, data is communicated between the processes by means of files. Each file contains all the instances of a particular object type (e.g., beams or columns). There is a one-to-one correspondence between the objects in the files and the individual process objects, although there are differences in format and attribute names. The data store manager is responsible for format and name translation and for transferring the appropriate files to and from the processes. As an illustration, Table 1 shows the ‘snapshot’ of a BEAM object before CONSTRUCTION PLANEX has supplied values for the last seven attributes.

In the current implementation, the data store manager maintains the global representation in the form of a global database schema implemented as a set of relations. The schema closely resembles the conceptual hierarchical tree representation and relies extensively on two types of relations: data relations which contain the attributes of the various objects, keyed by a single object-identifier attribute; and connection relations, usually all-key, which represent the part-of relations of the hierarchical tree.

In this implementation, the data store manager performs two distinct operations for each input and output transaction between a process and the data store. When providing input to a process, the manager first extracts an input view from the global schema through a sequence of relational operators, so that all information needed by the process is contained in the view; and second, makes the view available to the process. Two options are available to the processes: one is a flat file identical to that of the initial implementation; and the other is a file of database commands which, when executed by a copy of the DBMS manager on the process machine, creates a local copy of the view. The former option allows the
Table 1. BEAM instance data store object

<table>
<thead>
<tr>
<th>Slot</th>
<th>Value</th>
<th>Placed by</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>BEAM-378</td>
<td>STANLAY</td>
<td>SPEX PLANEX</td>
</tr>
<tr>
<td>CLASS</td>
<td>BEAM</td>
<td>STANLAY</td>
<td>SPEX PLANEX</td>
</tr>
<tr>
<td>IS-A</td>
<td>DE-GROUP</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>NAME</td>
<td>BEAM-11</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>NAME-CODE</td>
<td>81</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>PART-OF</td>
<td>TYP-FRAME-1540</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>MPOS</td>
<td>5273.4375</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>MNEG</td>
<td>0</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>VMAX</td>
<td>4</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>MULTIPLIER</td>
<td>4</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>UNBRACED-LENGTH</td>
<td>25.0</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>X1</td>
<td>(...)</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>X2</td>
<td>(...)</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>Y1</td>
<td>(...)</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>Y2</td>
<td>(...)</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>Z1</td>
<td>3</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>Z2</td>
<td>19</td>
<td>STANLAY</td>
<td>PLANEX</td>
</tr>
<tr>
<td>CONSTRUCTION-TYPE</td>
<td>NIL</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>DESIGNATION</td>
<td>W12 106</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>106</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>CONCRETE-TYPE</td>
<td>NIL</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>PSTEEL</td>
<td>NIL</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>GRADE</td>
<td>36</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>SHAPE</td>
<td>W</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>XL-DIMENSION</td>
<td>25.0</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>YL-DIMENSION</td>
<td>1.0183333</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>ZL-DIMENSION</td>
<td>1.0741667</td>
<td>SPEX</td>
<td>PLANEX</td>
</tr>
<tr>
<td>XG-COORDINATE</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
<tr>
<td>YG-COORDINATE</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
<tr>
<td>ZG-COORDINATE</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
<tr>
<td>MATERIAL-COST</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
<tr>
<td>CREW-COST</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
<tr>
<td>START-TIME</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
<tr>
<td>FINISH-TIME</td>
<td>NIL</td>
<td>PLANEX</td>
<td>USER</td>
</tr>
</tbody>
</table>

processes to run without modification, whereas the latter option is made available to processes which may wish to access the data through their own DBMS queries.

In receiving output from a process, the manager first receives the output view of the process, again by either of the two options; and second merges the output view into the global schema through a sequence of relational operators.

The only major conceptual difference between the two implementations is the explicit separation of input and output views. The input view is provided in a ‘read only’ mode: it is not recovered from the process upon its completion. In this fashion, database integrity maintenance is shared between the processes and the data store manager: the processes ensure that their output is consistent with respect to the input provided; while the data store manager ensures consistency among the outputs, i.e. consistency of the global representation.

CONTROL

Objectives

The controller is responsible for activating the individual processes. The design and subsequent evolution of the controller were dictated by three major considerations.

Support of different strategies. From the inception of the project, it was intended that the controller be able to support a number of different activation strategies, ranging from simple, fixed scheduling through demand-driven scheduling to a variety of planner- and blackboard-based opportunistic strategies.

Flexibility of implementation. Like the datastore, the controller was intended to have an evolution of its own, without affecting the individual processes. The Design Systems Laboratory of the Engineering Design Research Center at CMU, within which the IBDE project is conducted, has an active research program on generic design environments. The IBDE project is viewed by that Laboratory as a testbed for exploring the applicability of these environments in a specific application area. Therefore, flexibility in porting IBDE from one environment to another was a major consideration.

Minimal domain knowledge. To support the above two objectives, it is essential that the controller be as generic as possible, that is, that it require minimum...
knowledge about the domain of building design and construction planning. More specifically, it is desired that the controller "know" as little about the individual processes as possible, but use the messages on the blackboards and a static description of each process as guides.

Communication with processes

The processes communicate with the controller by means of blackboard messages. Each process is in one of three states: pending, active, or completed. Whenever an active process terminates, it sends a new status message which the controller posts on the status blackboard. The message signifies, in addition, whether the process was successful in producing a feasible solution or not. As discussed in the preceding section, the controller also controls the data communication between processes.

Implementations

The initial implementation provides a very limited control strategy, namely, an event-driven, sequential process activation. The controller maintains only the following static description about each process: the preconditions for its execution, namely, the process(es) that must have been successfully completed before the current process can be activated; and the machine on which the process runs. When the preconditions of a process are satisfied, the controller causes that process to be activated.

The controller is implemented on top of the DPSK (Distributed Problem Solving Kernel) system developed at CMU. DPSK provides an environment for distributed problem solving on multiple machines by programs written in different languages. It offers utilities for sending messages and signals between processes running on different machines; for generating and responding to events; and for communication between processes by means of a shared memory accessible to all the processes. DPSK was designed to facilitate the implementation of a variety of cooperative problem-solving architectures; the first IBDE implementation, with fixed precedence ordering between processes, was a relatively simple application of DPSK.

This implementation has been useful in bringing the first version of IBDE up to operational status. Changes in the processes, such as the addition of CORE and the replacement of the original HI-RISE process by STRYPES and STANLAY, were readily accommodated. The integrated system has been run on as many as 9 machines working simultaneously. The machines include HP-9000/320, MicroVAX, Sun 3 and Sun 4 systems. The controller and data store manager reside on a Sun 4 and the processes on Hewlett Packard workstations and Sun 3s except for SPEX. For reasons of efficiency, three copies of SPEX reside on three MicroVAXes and process, respectively, the bottom storey columns (so as to supply input to FOOTER quickly), the remaining columns, and the other structural components.

The authors are in the process of designing the second implementation of the controller. It will handle the expanded processes discussed in the second section by acting on the design constraints posted on the message blackboard. A variety of activation strategies will be explored, including both a planner and an opportunistic scheduler. Two generic design environments developed by EDRC are being considered as the basis for implementation.

USER INTERFACES

An integrated software environment for building design and construction is incomplete and inaccessible without a common user display interface. As the interface is intended for a variety of users with different backgrounds, it must conform to certain graphical standards and should exhibit a degree of intelligence. An interface of this type was developed for the IBDE project. It provides a uniform set of interface facilities for the following functions:

- Graphical display of the status of all processes. Each process is shown as either active, completed, or acting as a critic to another process.
- Graphical display of data at any level of the project data store representation. As soon as a process is completed, the content of the project data store can be displayed. The designer sees the geometric representation of these data as 3D objects or as charts and symbols. As an example, Figure 4 shows how the interface presents a portion of the schedule computed by CONSTRUCTION PLANEX.
- Textual and graphical display of object classes. The user selects one of the data store objects directly from a menu, and the geometric and textual information is displayed.
- Graphical display of selected items. The designer can specify constraints to view objects of a certain class or that fall within user-defined limits. All objects found conforming to the constraints are highlighted on the graphical display.
- Graphical navigation to select specific objects. Once selected graphically, the object is highlighted and the appropriate database information appears on screen in a pop-up window.
- Animation of the construction process. After the necessary data are available in the data store, an animation of the construction process may be selected. This option is under development.

In addition, each process has its own interface for:

- Run-time interactions with the process, including the controller, to provide process-specific input, interactions and user decisions. Examples are to select a structural system alternative for STRYPES for further processing or to override the controller's agenda.
- Run-time process display. This includes process-specific information not accessible through the common user display interface, such as the passing of messages in the ARCHPLAN interface.

A long-term goal in the development of the common user display interface is to increase understanding of
different processes and their interaction for engineers and designers. It is hoped that the outcome will not only encourage a sharing of design ideas and the direct graphical and numerical monitoring of their consequences, but also improve the quality of the designed product.

CONCLUSIONS

IBDE is a testbed for the exploration of integration and communication issues in the building industry. The processes involved are knowledge-based and can serve as surrogate experts (provided, of course, that they adequately capture and use the expertise relevant to their particular task). This enables the running of a variety of experiments when exploring a particular issue, so that the conclusions reached will have a strong empirical basis. Furthermore, the modular nature of the environment provides a testbed for the empirical evaluation and calibration of generic integrated design support environments. Experiments with these environments provide feedback to their developers and can eventually lead to extrapolations to other design disciplines.

The authors deliberately did not start with an overall, normative model of the building design process. Rather, they prefer to arrive at generalizations about this process based on the experience and insights gained from their experiments. The environment itself is intended to serve as a vehicle for discovery and theory formation.

It would be premature to speculate what a ‘production’ version of IBDE may look like. The project must first discover a ‘language’ through which the project participants’ intent may be communicated to others, and through which critiques can be fed back. Even then it remains to be seen how the resulting intimate integration can be implemented in the present dispersed organizational structure of the industry.

REFERENCES

1 Hendrickson, C and Au, T Project Management for Construction Prentice-Hall (1989)


